

Symplectic finite-difference methods for solving partial differential equations

Siu A. Chin

Department of Physics, Texas A&M University, College Station, TX 77843, USA

The usual explicit finite-difference method of solving partial differential equations is limited in stability because it approximates the exact amplification factor by power-series. By adapting the same exponential-splitting method of deriving symplectic integrators, explicit symplectic finite-difference methods produce Saul'yev-type schemes which approximate the exact amplification factor by rational-functions. As with conventional symplectic integrators, these symplectic finite-difference algorithms preserve important qualitative features of the exact solution. Thus the symplectic diffusing algorithm is *unconditionally stable* and the symplectic advection algorithm is *unitary*. There is a one-to-one correspondence between symplectic integrators and symplectic finite-difference methods, including the key idea that one can systematically improve an algorithm by matching its modified Hamiltonian more closely to the original Hamiltonian. Consequently, the entire arsenal of symplectic integrators can be used to produce arbitrary high order time-marching algorithms for solving the diffusion and the advection equation.

I. INTRODUCTION

The 1D diffusion equation

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} \quad (1.1)$$

can be solved numerically by applying the forward-time and central-difference approximations to yield the explicit algorithm

$$u'_j = u_j + r[(u_{j+1} - u_j) - (u_j - u_{j-1})], \quad (1.2)$$

where $x_j = j\Delta x$, $u_j = u(x_j, t)$, $u'_j = u(x_j, t + \Delta t)$ and

$$r = \frac{\Delta t D}{\Delta x^2}. \quad (1.3)$$

Under this (Euler) algorithm, each Fourier component $\tilde{u}_k = e^{ikx}$ with wave number k is amplified by a factor of

$$g = 1 - 4r \sin^2(k\Delta x/2), \quad (1.4)$$

restricting stability ($|g| \leq 1$) to the Courant-Friedrichs-Lewy¹ (CFL) limit,

$$r \leq \frac{1}{2}. \quad (1.5)$$

Since explicit finite-difference methods approximate the exact amplification factor by power-series such as (1.4), it seems inevitable that they will eventually blow-up and be limited in stability. However, Saul'yev^{2,3} showed in the 50's that, by simply replacing in (1.2), either

$$(u_j - u_{j-1}) \rightarrow (u'_j - u'_{j-1}) \quad \text{or} \quad (u_{j+1} - u_j) \rightarrow (u'_{j+1} - u'_j) \quad (1.6)$$

one would have *unconditionally stable* algorithms:

$$u'_j = \beta_S u'_{j-1} + \gamma_S u_j + \beta_S u_{j+1}, \quad (1.7)$$

or

$$u'_j = \beta_S u_{j-1} + \gamma_S u_j + \beta_S u'_{j+1}, \quad (1.8)$$

where γ_S and β_S are Saul'yev's coefficients given by

$$\gamma_S = \frac{1-r}{1+r} \quad \text{and} \quad \beta_S = \frac{r}{1+r}. \quad (1.9)$$

Algorithm (1.7) is explicit if it is evaluated in ascending order in j from left to right and if the left-most u_1 is a boundary value fixed in time. Similarly, algorithm (1.8) is explicit if it is evaluated in descending order in j from right to left and if the right-most u_N is a boundary value fixed in time. Saul'yev also realized that both algorithms have large errors (including phase errors due to their asymmetric forms), but if they are applied *alternately*, the error would be greatly reduced after such a pair-wise application. This then gives rise to *alternating direction explicit* algorithms advocated by Larkin⁴ and generalized to *alternating group explicit* algorithms by Evans^{5,6}.

There are four unanswered questions about Saul'yev asymmetric algorithms: 1) While it is easy to show that algorithm (1.7) and (1.8) are unconditionally stable, there is no deeper understanding of this stability. 2) The algorithms are not explicit in the case of periodic boundary. What would be the algorithm if there are no fixed boundary values? 3) The alternating application of (1.7) and (1.8) greatly reduces the resulting error. How can one characterize this improvement precisely? 4) How can Saul'yev-type algorithms be generalized to higher orders?

This work presents a new way of deriving finite-difference schemes based on exponential-splittings rather than Taylor expansions. Exponential-splitting is the basis for developing symplectic integrators⁷⁻¹³, the hallmark of structure-preserving algorithms. The finite-difference method presented here is properly “symplectic” in the original sense that it has certain “intertwining” quality, resembling Hamilton’s equations. It is also symplectic in the wider sense of structure-preserving, in that there is a Hamiltonian-like quantity that the algorithms seek to preserve.

As will be shown, there is a one-to-one correspondence between symplectic finite-difference methods and symplectic integrators. It is therefore useful to summarize some basic results of symplectic integrators for later reference. Symplectic integrators are based on approximating $e^{\epsilon(\mathbf{A}+\mathbf{B})}$ to any order in ϵ via a single product decomposition

$$e^{\epsilon(\mathbf{A}+\mathbf{B})} = \prod_i e^{a_i \epsilon \mathbf{A}} e^{b_i \epsilon \mathbf{B}}, \quad (1.10)$$

where \mathbf{A} and \mathbf{B} are non-commuting operators (or matrices). Usually, $\mathbf{A}+\mathbf{B}=\mathbf{H}$ is the Hamiltonian operator and $e^{\epsilon \mathbf{H}}$ is the evolution operator that evolves the system forward for time ϵ . The key idea is to preserve the *exponential* character of the evolution operator. The two first-order, Trotter¹⁴ approximations are

$$T_{1A}(\epsilon) = e^{\epsilon \mathbf{A}} e^{\epsilon \mathbf{B}}, \quad T_{1B}(\epsilon) = e^{\epsilon \mathbf{B}} e^{\epsilon \mathbf{A}}, \quad (1.11)$$

and the two second-order Strang¹⁵ approximations are

$$\begin{aligned} T_{2A}(\epsilon) &= T_{1A}(\epsilon/2) T_{1B}(\epsilon/2) = e^{\frac{1}{2}\epsilon \mathbf{A}} e^{\epsilon \mathbf{B}} e^{\frac{1}{2}\epsilon \mathbf{A}}, \\ T_{2B}(\epsilon) &= T_{1B}(\epsilon/2) T_{1A}(\epsilon/2) = e^{\frac{1}{2}\epsilon \mathbf{B}} e^{\epsilon \mathbf{A}} e^{\frac{1}{2}\epsilon \mathbf{B}}. \end{aligned} \quad (1.12)$$

The approximation

$$T_{2C}(\epsilon) = \frac{1}{2} [T_{1A}(\epsilon) + T_{1B}(\epsilon)] \quad (1.13)$$

is also second-order, but since it is no longer a single product of exponentials, it is no longer symplectic. In most cases, it is inferior to T_{2A} and T_{2B} because the time steps used in evaluating T_{1A} and T_{1B} are twice as large as those used in T_{2A} and T_{2B} .

Let T_2 denotes either T_{2A} or T_{2B} . T_2 must be second order because for a left-right symmetric product as above, it must obey

$$T_2(-\epsilon) T_2(\epsilon) = 1, \quad (1.14)$$

and therefore must be of the form,

$$T_2(\epsilon) = e^{\epsilon \mathbf{H} + \epsilon^3 \mathbf{E}_3 + \epsilon^5 \mathbf{E}_5 + \dots} \quad (1.15)$$

with only odd powers of ϵ in the exponent. (Since there is no way for the operators in (1.14) to cancel if there are any even power terms in ϵ .) In (1.15), \mathbf{E}_n denote higher order commutators of \mathbf{A} and \mathbf{B} . The algorithm corresponding to T_2 then yields exact trajectories of the second-order *modified Hamiltonian*

$$\mathbf{H}_2(\epsilon) = \mathbf{H} + \epsilon^2 \mathbf{E}_3 + \epsilon^4 \mathbf{E}_5 + \dots \quad (1.16)$$

A standard way of improving the efficiency of symplectic integrators is to generate a $2n^{th}$ -order algorithm via a product of second-order algorithms^{7,9,10}, via

$$T_{2n}(\epsilon) = \prod_{i=1}^N T_2(a_i \epsilon). \quad (1.17)$$

Since the error structure of $T_2(\epsilon)$ is given by (1.15), to preserve the original Hamiltonian, one must choose a_i to preserve the first power of ϵ ,

$$\sum_{i=1}^N a_i = 1. \quad (1.18)$$

To obtain a fourth-order algorithm, one must eliminate the error term proportional to ϵ^3 by requiring,

$$\sum_{i=1}^N a_i^3 = 0. \quad (1.19)$$

For a sixth-order algorithm, one must require the above and

$$\sum_{i=1}^N a_i^5 = 0, \quad (1.20)$$

and so on. While proofs of these assertions in terms of operators are not difficult, we will not need them. Symplectic finite-difference methods use a much simpler version of these ideas. Instead of dealing with the evolution operator $e^{\mathbf{H}}$, the finite difference method has a proxy, the amplification factor, which is just a function. Order-conditions such as (1.19) and (1.20) will then be obvious. Other results will be cited as needed, but these basic findings are sufficient to answer the four questions about Saul'yev's schemes. For the next two sections we will give a detailed derivation of the symplectic diffusion and advection algorithms, followed by a discussion of the diffusion-advection equation and a concluding summary.

II. SYMPLECTIC DIFFUSION ALGORITHM

Consider solving the diffusion equation (1.1) with periodic boundary condition $u_{N+1} = u_1$ in the semi-discretized form,

$$\frac{du_j}{dt} = \frac{D}{\Delta x^2} (u_{j+1} - 2u_j + u_{j-1}). \quad (2.1)$$

Regarding u_j as a vector, this is

$$\frac{d\mathbf{u}}{dt} = \mathbf{A}\mathbf{u}, \quad (2.2)$$

with

$$\mathbf{u} = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}, \quad \mathbf{A} = \frac{D}{\Delta x^2} \begin{pmatrix} -2 & 1 & & 1 \\ 1 & -2 & 1 & \\ & & \ddots & \\ & & 1 & -2 & 1 \\ 1 & & & 1 & -2 \end{pmatrix}, \quad (2.3)$$

and exact solution

$$\mathbf{u}(t + \Delta t) = e^{\Delta t \mathbf{A}} \mathbf{u}(t). \quad (2.4)$$

The Euler algorithm corresponds to expanding out the exponential to first order in Δt

$$\mathbf{u}(t + \Delta t) = (1 + \Delta t \mathbf{A}) \mathbf{u}(t), \quad (2.5)$$

resulting in a power-series amplification factor (1.4), with limited stability.

If the exponential in (2.4) can be solved exactly, the amplification factor would be

$$g_{ex} = e^{-h_{ex}}, \quad (2.6)$$

where

$$h_{ex} = r4\sin^2(\theta/2) \quad \text{and} \quad \theta \equiv k\Delta x. \quad (2.7)$$

The *amplification exponent* h_{ex} here plays the role of a time parameter r times the original “Hamiltonian” $h_0 = 4\sin^2(\theta/2)$. The resulting algorithm will then be *unconditionally stable* for all $r > 0$. In the limit of $\Delta x \rightarrow 0$, each k -Fourier components will be damped by $g_{ex} = e^{-\Delta t D k^2}$, which is the exact solution to (1.1).

To preserve this important feature of the exact solution, one must seek alternative ways of approximating of $e^{\Delta t \mathbf{A}}$ without doing any Taylor expansion. The structure of \mathbf{A} immediately suggests that it should decompose as

$$\mathbf{A} = \sum_{j=1}^N \mathbf{A}_j, \quad (2.8)$$

where each \mathbf{A}_j has only a single, non-vanishing 2×2 matrix along the diagonal connecting the j and the $j+1$ elements:

$$\mathbf{A}_j = \frac{D}{\Delta x^2} \begin{pmatrix} & & & \\ & \ddots & & \\ & & -1 & 1 \\ & & 1 & -1 \\ & & & \ddots \end{pmatrix} \quad \text{and} \quad \mathbf{A}_N = \frac{D}{\Delta x^2} \begin{pmatrix} -1 & & & 1 \\ & \ddots & & \\ & & \ddots & \\ 1 & & & -1 \end{pmatrix}. \quad (2.9)$$

The exponential of each \mathbf{A}_j can now be evaluated exactly:

$$e^{\Delta t \mathbf{A}_j} = \begin{pmatrix} 1 & & & \\ \alpha & \beta & & \\ \beta & \alpha & & \\ & & & 1 \end{pmatrix}, \quad e^{\Delta t \mathbf{A}_N} = \begin{pmatrix} \alpha & & \beta \\ & 1 & \\ & & 1 \\ \beta & & \alpha \end{pmatrix}, \quad (2.10)$$

where

$$\alpha = \frac{1}{2}(1 + \gamma), \quad \beta = \frac{1}{2}(1 - \gamma), \quad \text{and} \quad \gamma = e^{-2r}. \quad (2.11)$$

Each $e^{\Delta t \mathbf{A}_j}$ updates only u_j and u_{j+1} as

$$\begin{aligned} u'_j &= \alpha u_j + \beta u_{j+1} \\ u'_{j+1} &= \beta u_j + \alpha u_{j+1}. \end{aligned} \quad (2.12)$$

The eigenvalues of this updating matrix are $\alpha \pm \beta = 1$, γ , with $\det = \gamma$. This means that the updating is dissipative for $r > 0$ and unstable for $r < 0$. Since α and β are given in terms of γ , *the resulting algorithm depends only on a single parameter γ* .

One can now decompose $\exp(\Delta t \mathbf{A})$ to first order in Δt (apply (1.11) repeatedly) via either

$$T_{1A}(\Delta t) = e^{\Delta t \mathbf{A}_N} \dots e^{\Delta t \mathbf{A}_2} e^{\Delta t \mathbf{A}_1}, \quad (2.13)$$

or

$$T_{1B}(\Delta t) = e^{\Delta t \mathbf{A}_1} \dots e^{\Delta t \mathbf{A}_{N-1}} e^{\Delta t \mathbf{A}_N}. \quad (2.14)$$

These algorithms update the grid points sequentially, two by two at a time according to (2.12), but each grid point is updated *twice*, in an intertwining manner. This is crucial for dealing with the periodic boundary condition. Let u_j^* denotes the first time when u_j is updated and u'_j the second (and final) time it is updated. One then has for algorithm 1A:

$$\begin{aligned} u_1^* &= \alpha u_1 + \beta u_2 \\ u_2^* &= \beta u_1 + \alpha u_2 \\ u'_2 &= \alpha u_2^* + \beta u_3 \\ u_3^* &= \beta u_2^* + \alpha u_3. \\ &\dots \\ u'_j &= \alpha u_j^* + \beta u_{j+1} \\ u_{j+1}^* &= \beta u_j^* + \alpha u_{j+1}. \\ &\dots \\ u'_N &= \alpha u_N^* + \beta u_1^* \\ u'_1 &= \beta u_N^* + \alpha u_1^*. \end{aligned} \quad (2.15)$$

$$(2.16)$$

Since $\alpha + \beta = 1$, summing up both sides from (2.15) to (2.16) gives,

$$\sum_{j=1}^N u'_j = \sum_{j=1}^N u_j. \quad (2.17)$$

The algorithm is therefore norm-conserving. The same is true of algorithm 1B below. For $2 < j < N$ one has

$$\begin{aligned} u'_j &= \alpha u_j^* + \beta u_{j+1} \\ &= \alpha(\beta u_{j-1}^* + \alpha u_j) + \beta u_{j+1} \\ &= \beta(u'_{j-1} - \beta u_j) + \alpha^2 u_j + \beta u_{j+1} \\ &= \beta u'_{j-1} + \gamma u_j + \beta u_{j+1} \end{aligned} \quad (2.18)$$

and for $j = 2, N$,

$$\begin{aligned} u'_2 &= \beta u_1^* + \gamma u_2 + \beta u_3 \\ u'_N &= \beta u_{N-1}^* + \gamma u_N + \beta u_1^*. \end{aligned} \quad (2.19)$$

Finally when the snake bits its tail, one has

$$u'_1 = \frac{\beta}{\alpha} u'_N + \gamma u_1 + \frac{\gamma\beta}{\alpha} u_2. \quad (2.20)$$

Similarly, 1B is given by

$$u_1^* = \beta u_N + \alpha u_1 \quad (2.21)$$

$$\begin{aligned} u'_N &= \beta u_{N-1} + \gamma u_N + \beta u_1^* \\ u'_j &= \beta u_{j-1} + \gamma u_j + \beta u'_{j+1} \end{aligned} \quad (2.22)$$

$$\begin{aligned} u'_2 &= \beta u_1^* + \gamma u_2 + \beta u'_3 \\ u'_1 &= \frac{\gamma\beta}{\alpha} u_N + \gamma u_1 + \frac{\beta}{\alpha} u'_2. \end{aligned} \quad (2.23)$$

Algorithms 1A and 1B are essentially given by (2.18) and (2.22) respectively, except for three values of u'_1 , u'_2 and u'_N . The forms of (2.18) and (2.22) reproduce Saul'yev's schemes (1.7) and (1.8), but with different coefficients. Saul'yev's coefficient γ_S is a rational approximation to the $\gamma = e^{-2r}$ here. Note that his $\beta_S = r/(1+r)$ is also given by $\beta_S = (1 - \gamma_S)/2$. In contrast to Saul'yev's algorithm, which cannot be started for periodic boundary condition, algorithm 1A and 1B are truly explicit because they are fundamentally given by the sequential updating of (2.12). Each algorithm can get started by first updating u_1 to u_1^* , then updating it again at the end to u'_1 .

By virtue of (1.12) one can now immediately generate a second-order time-marching algorithm via the symmetric product,

$$\begin{aligned} T_2(\Delta t) &= T_{1B}\left(\frac{\Delta t}{2}\right) T_{1A}\left(\frac{\Delta t}{2}\right) \\ &= e^{\frac{1}{2}\Delta t \mathbf{A}_1} e^{\frac{1}{2}\Delta t \mathbf{A}_2} \dots e^{\frac{1}{2}\Delta t \mathbf{A}_N} e^{\frac{1}{2}\Delta t \mathbf{A}_N} \dots e^{\frac{1}{2}\Delta t \mathbf{A}_2} e^{\frac{1}{2}\Delta t \mathbf{A}_1}. \end{aligned} \quad (2.24)$$

If the boundary effects of u'_1 , u'_2 and u'_N are ignored (for now) and 1A and 1B are considered as given by (2.18) and (2.22), then the alternative product $T_{1A}(\Delta t/2) T_{1B}(\Delta t/2)$ yields the same second-order algorithm. In this case, algorithms 1A and 1B have amplification factors

$$\begin{aligned} g_{1A} &= \frac{\gamma + \beta e^{i\theta}}{1 - \beta e^{-i\theta}}, \\ g_{1B} &= \frac{\gamma + \beta e^{-i\theta}}{1 - \beta e^{i\theta}}, \end{aligned} \quad (2.25)$$

with opposite phase errors, and the second-order algorithm has

$$\begin{aligned} g_2 &= g_{1B} \left(\frac{\Delta t}{2} \right) g_{1A} \left(\frac{\Delta t}{2} \right) \\ &= \frac{\tilde{\gamma}^2 + \tilde{\beta}^2 + 2\tilde{\beta}\tilde{\gamma} \cos\theta}{1 + \tilde{\beta}^2 - 2\tilde{\beta}\tilde{\gamma} \cos\theta}, \end{aligned} \quad (2.26)$$

$$= \frac{1 - (4\tilde{\beta}\tilde{\gamma}/\tilde{\alpha}^2) \sin^2\theta/2}{1 + (4\tilde{\beta}/\tilde{\alpha}^2) \sin^2\theta/2} = e^{-h_2}, \quad (2.27)$$

with no phase error and where

$$\tilde{\alpha} = \frac{1}{2}(1 + \tilde{\gamma}), \quad \tilde{\beta} = \frac{1}{2}(1 - \tilde{\gamma}) \quad \text{and} \quad \tilde{\gamma} = \gamma(r/2). \quad (2.28)$$

Since both algorithms 1A and 1B have phase errors, only the second-order algorithm is qualitatively similar to the exact solution. Eq.(2.27) makes it clear that this algorithm is unconditionally stable since $0 \leq \tilde{\gamma} \leq 1$. Algorithms 1A and 1B are also unconditionally stable since $|g_{1A,1B}| = \sqrt{g_2}$ with $\tilde{\gamma} \rightarrow \gamma$. Note that this also proves the unconditional stability of Saul'yev's algorithms. His coefficient γ_S can turn negative, but only approaches -1 as $r \rightarrow \infty$. Conventional explicit methods, like that of the Euler algorithm, are limited in stability because they have power-series amplification factors. By contrast, symplectic finite-difference methods are unconditionally stable because they produce Saul'yev-type schemes with *rational-function* amplification factors. This type of stability is usually associated only with *implicit* methods.

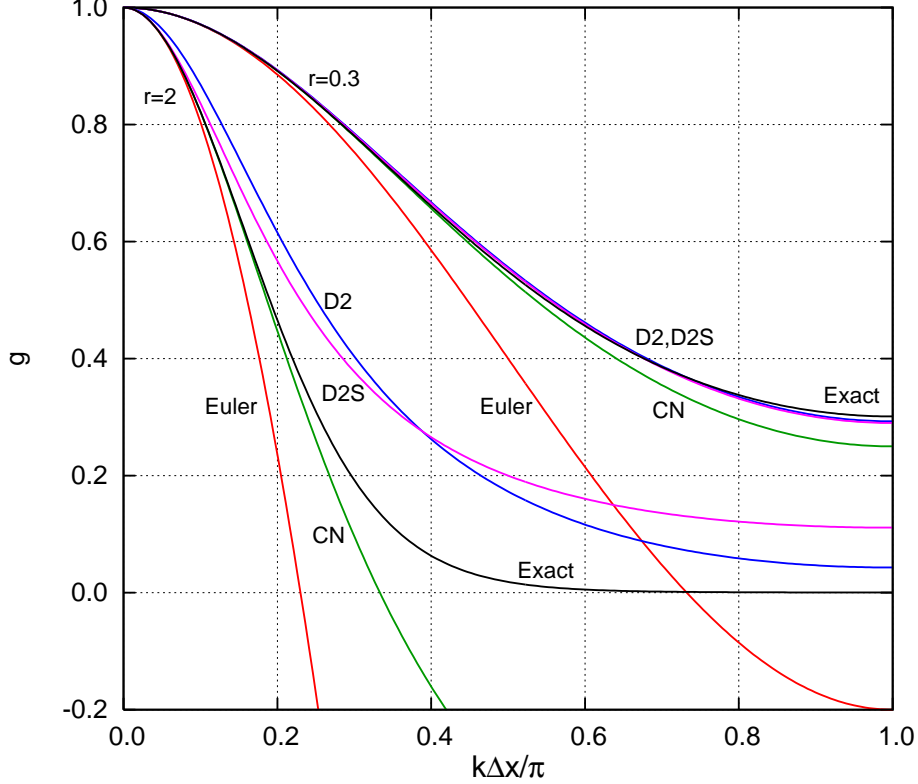


FIG. 1: Comparing the amplification factor of various diffusion algorithms at $r = 0.3$ and $r = 2$. Euler is the first-order explicit algorithm. CN is the second-order implicit Crank-Nicolson algorithm. D2 is the second-order symplectic algorithm using the original coefficients (2.28) and D2S is the same algorithm but uses Saul'yev's coefficient (2.33). "Exact" is g_{ex} of (2.6).

Since the order of matrices defining $T_2(\Delta t)$ in (2.24) is left-right symmetric, one has the same situation as in the symplectic integrators case of (1.14), implying that $g_2(-r)g_2(r) = 1$. This means that $g_2(-r) > 1$ for all $k \neq 0$ modes and the algorithm is unconditionally unstable for negative time steps. Moreover, this also means that $h_2(-r) = -h_2(r)$ and $h_2(r)$ is an odd function of r . This is a simpler, functional version of (1.15). Expanding h_2 of (2.27) in powers of θ gives,

$$h_2 = \frac{2(1 - \tilde{\gamma})}{1 + \tilde{\gamma}}\theta^2 - \frac{(13 - 35\tilde{\gamma} + 35\tilde{\gamma}^2 - 13\tilde{\gamma}^3)}{6(1 + \tilde{\gamma})^3}\theta^4 + \dots \quad (2.29)$$

Each coefficient must be an odd function of r . This is satisfied only if

$$\tilde{\gamma}(-r)\tilde{\gamma}(r) = 1. \quad (2.30)$$

Thus every function $\tilde{\gamma}(r)$ satisfying (2.30) with $|\tilde{\gamma}(r)| \leq 1$ defines an unconditionally stable algorithm for solving the

diffusion equation. For the original choice of $\tilde{\gamma}(r) = e^{-r}$, one finds

$$h_2 = (r - \frac{r^3}{12} + \frac{r^5}{120} \dots) \theta^2 - (\frac{r}{12} + \frac{35r^3}{144} + \dots) \theta^4 + (\frac{r}{360} + \frac{539r^3}{4320} + \dots) \theta^6 + \dots \quad (2.31)$$

Comparing this to the expansion of the exact amplification exponent,

$$h_{ex} = r\theta^2 - \frac{r}{12}\theta^4 + \frac{r}{360}\theta^6 + \dots, \quad (2.32)$$

one sees that the original choice does not reproduce leading term exactly except when $r \ll 1$. To improve this, let's take $\tilde{\gamma}(r)$ to be an arbitrary function of r but with $\tilde{\alpha}$ and β still defined by (2.28). The first term in h_{ex} can now be matched exactly by requiring

$$\frac{1 - \tilde{\gamma}(r)}{1 + \tilde{\gamma}(r)} = \frac{r}{2} \quad \rightarrow \quad \tilde{\gamma}(r) = \frac{1 - r/2}{1 + r/2}, \quad (2.33)$$

which is precisely Saul'yev's original coefficient. With this choice for $\tilde{\gamma}(r)$, (2.27) reads

$$g_2 = \frac{1 - 2r(1 - r/2) \sin^2(\theta/2)}{1 + 2r(1 + r/2) \sin^2(\theta/2)}, \quad (2.34)$$

with exponent

$$h_2 = r\theta^2 - (\frac{r}{12} + \frac{r^3}{4})\theta^4 + (\frac{r}{360} + \frac{r^3}{8} + \frac{r^5}{16})\theta^6 + \dots \quad (2.35)$$

which is now correct to third-order in θ . Comparing this and (2.31) to h_{ex} , one sees that all the error terms of h_2 are *odd* powers of r higher than the first. As a matter of fact, by resumming terms proportional to r , we can make this error structure in exact conformity with (1.15),

$$h_2 = rh_0 + r^3 E_3(\theta) + r^5 E_5(\theta) + \dots, \quad (2.36)$$

where $h_0 = 4 \sin^2(\theta/2)$. This is the same error structure exploited by symplectic integrators to produce higher order algorithms.

Saul'yev's algorithm is close in reproducing the amplification factor of the *implicit* Crank-Nicolson (CN) scheme (which is without the $\pm r/2$ terms in (2.34)). The CN scheme has the advantage that its exponent is

$$h_{CN} = r\theta^2 - \frac{r}{12}\theta^4 + (\frac{r}{360} + \frac{r^3}{12})\theta^6 + \dots \quad (2.37)$$

which is correct to fifth-order in θ . In Fig.1, we compare the amplification factor of various algorithms at two values of r . D2 and D2S are second-order symplectic diffusion algorithms described above using the original coefficient (2.28) and Saul'pev's coefficient (2.33), respectively. For $r = 0.3$, both D2 and D2S track g_{ex} closely over the entire range k values. Both Euler and CN tend to over-damp higher Fourier modes. At $r = 2.0$, while both Euler and CN turn negative at large k , D2 and D2S remain positive, like that of g_{ex} . At large r , D2S is clearly better than D2 at small k .

To generalize Saul'yev schemes to periodic boundary condition, one simply replaces in the above algorithms, $\gamma \rightarrow \gamma_S$. In Fig.2 we show the working of algorithms 1A, 1B and 2 using both sets of coefficients. With the original coefficients, the phase errors are much smaller, but the under-damp error is much larger. For Saul'pev's coefficient, the phase errors are much greater, but the under-damp error is smaller. Since the phase error is automatically eliminated by going to second-order, D2S has an advantage over D2.

If one were to construct a $2n^{th}$ -order algorithm out of a product of second-order algorithms

$$T_{2n}(\Delta t) = \prod_{i=1}^N T_2(a_i \Delta t), \quad (2.38)$$

then the corresponding h_{2n} is given by

$$\begin{aligned} h_{2n}(r) &= \sum_{i=1}^N h_2(a_i r) \\ &= rh_0 \sum_{i=1}^N a_i + r^3 E_3(\theta) \sum_{i=1}^N a_i^3 + r^5 E_5(\theta) \sum_{i=1}^N a_i^5 + \dots, \end{aligned} \quad (2.39)$$

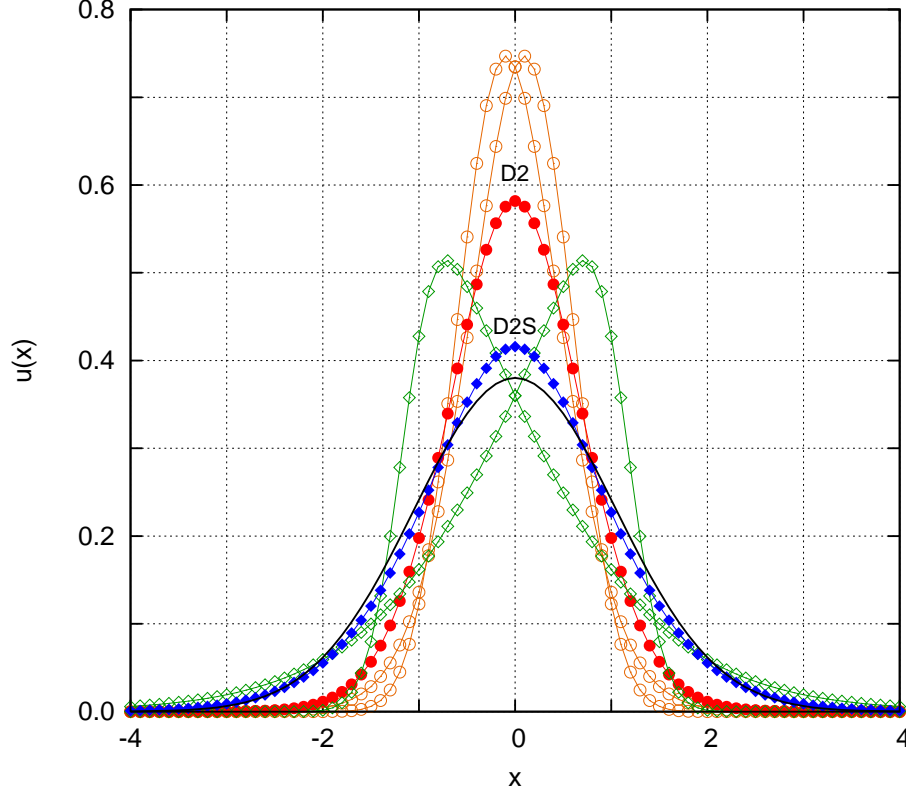


FIG. 2: The diffusion of a Gaussian profile after $t = 1$ on a grid of 120 points spanning the interval $[-6, 6]$ with $D = 1/2$ and $\Delta t = 0.1$, corresponding to $r = 5$. This large value is chosen to exaggerate various errors. The open and filled circles are algorithms 1A, 1B and 2 using the standard coefficients. The open and filled diamonds are the same three algorithms using Saul'pev's coefficients. The solid black line is the exact solution in the continuum limit.

and the order conditions (1.18)-(1.20) are easily understood. Unfortunately, for the diffusion algorithm, $T_2(a_i \Delta t)$ is unstable for any negative a_i and no negative coefficient a_i can be allowed. In this case, order-conditions such as (1.19) and (1.20) cannot be satisfied and no higher-order composition algorithms of the form (2.38) is possible. (However, these algorithms will be useful for solving the advection equation in the next section.)

To overcome this impasse, one must go beyond the single product approximation of (2.38), and consider a multi-product expansion of the form

$$T_{2n}(\Delta t) = \sum_k c_k \prod_{i=1}^{N_k} T_2(a_{ki} \Delta t). \quad (2.40)$$

If a_{ki} were to remain positive, then Shang¹⁶ has shown that any single product in (2.40) can at most be second order and that some c_k coefficients must be negative. More recently this author realized that^{17,18}, due to the error structure (2.36), the first-order term rh_0 is automatically preserved by monomial products of the form $T_2^k(\Delta t/k)$ with exponent

$$h_2^{(k)} = rh_0 + k^{-2}r^3E_3(\theta) + k^{-4}r^5E_5(\theta) + \dots \quad (2.41)$$

The arbitrariness in N_k and a_{ki} can be eliminated by taking $N_k = k$ and $a_{ki} = 1/k$. This then produces a much simpler Multi-Product Expansion¹⁷ (MPE)

$$T_{2n}(\Delta t) = \sum_k c_k T_2^k(\Delta t/k) \quad (2.42)$$

for any sequence of n whole numbers $\{k\}$ with analytically known coefficients c_k . For the harmonic sequence of $k = 1, 2, 3, \dots$, the first few higher order algorithms are:

$$T_4(\Delta t) = -\frac{1}{3}T_2(\Delta t) + \frac{4}{3}T_2^2\left(\frac{\Delta t}{2}\right) \quad (2.43)$$

$$T_6(\Delta t) = \frac{1}{24}T_2(\Delta t) - \frac{16}{15}T_2^2\left(\frac{\Delta t}{2}\right) + \frac{81}{40}T_2^3\left(\frac{\Delta t}{3}\right) \quad (2.44)$$

$$T_8(\Delta t) = -\frac{1}{360}T_2(\Delta t) + \frac{16}{45}T_2^2\left(\frac{\Delta t}{2}\right) - \frac{729}{280}T_2^3\left(\frac{\Delta t}{3}\right) + \frac{1024}{315}T_2^4\left(\frac{\Delta t}{4}\right). \quad (2.45)$$

For the diffusion equation, the use of the fourth-order extrapolation (2.43) has been previously suggested by Schatzman¹⁹. However, these high order methods are useless for conventional explicit schemes, since they are unstable at large time steps. It is only with the use of unconditionally stable algorithms here that the power of these high order schemes can be unleashed. These MPE algorithms do not preserve the positivity of the initial profile. This is in keeping with the general observation that there can't be any finite-difference scheme for solving the diffusion equation that preserves positivity beyond the second-order²¹. More recently, Zillich, Mayrhofer and Chin²² have shown that Path-Integral Monte Carlo simulations, where positivity is of the utmost importance, can be successfully carried out using these expansions, demonstrating that the violation of positivity is small and controllable. These MPE algorithms are also not symplectic. However, as argued by Blanes, Casas and Ros²⁰, they are symplectic to order $2n + 3$. Thus at sufficiently high orders, they are indistinguishable from truly symplectic algorithms up to machine precision. In the context of classical dynamics, these MPE algorithms have been tested up to the 100th order in Ref.18.

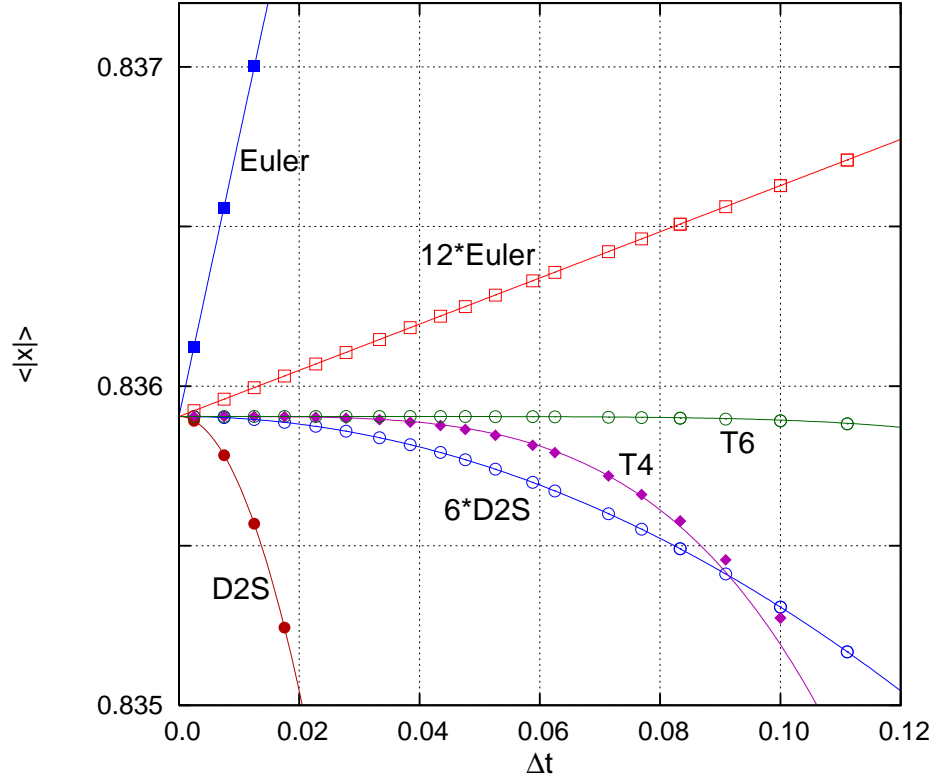


FIG. 3: The convergence of $\langle |x| \rangle$ after a Gaussian profile has been diffused for $t = 1$ on a grid of 120 points spanning the interval $[-6, 6]$ with $D = 1/2$. The range of Δt , from $\Delta t = 1/400$ to $\Delta t = 1/9$, corresponds to a range of $r = 0.125$ to $r = 5.555$. Lines are fitted power laws Δt^n verifying the order of the algorithm. “12Euler” labels results of running the first order Euler algorithm 12 times at time step $\Delta t/12$. “6D2S” are results from running the second-order algorithm D2S six times at time step $\Delta t/6$. T_4 and T_6 are fourth and sixth-order algorithms which require 3 and 6 runs of D2S respectively.

In Fig.3, we compare and verify the order of convergence of T_4 and T_6 with D2S as T_2 by computing the expectation value

$$\langle |x| \rangle = \frac{\sum_{j=1}^N |j \Delta x| u_j}{\sum_{j=1}^N u_j} \quad (2.46)$$

after evolving u_j for $t = 1$ as a function of Δt . The absolute value is used because the Euler algorithm would exactly preserve $\langle x \rangle$ even when it is unstable. For computing $\langle |x| \rangle$, the Euler algorithm is extremely linear within its tiny range of stability. Since it tends to over-damp, its evolving profile is flatter and $\langle |x| \rangle$ converges from above. Symplectic diffusion algorithms under-damp, and their results for $\langle |x| \rangle$ converge from below. All results can be well fitted with power laws of the form $a + b\Delta t^n$ with $n = 1, 2, 4$ and 6 , verifying the order of the algorithms. To show that these higher order algorithms are more efficient than running low order algorithms at reduced step sizes, we also plotted results of running the Euler algorithm 12 times at time step $\Delta t/12$ and algorithm D2S six times at $\Delta t/6$.

III. SYMPLECTIC ADVECTION ALGORITHM

For the advection equation

$$\frac{\partial u}{\partial t} = -v \frac{\partial u}{\partial x}, \quad (3.1)$$

its usual semi-discrete form is

$$\frac{\partial u_j}{\partial t} = -\frac{v}{2\Delta x}(u_{j+1} - u_{j-1}), \quad (3.2)$$

with discretization matrix

$$\mathbf{B} = \frac{v}{2\Delta x} \begin{pmatrix} 0 & -1 & & 1 \\ 1 & 0 & -1 & \\ & & \ddots & \\ & & 1 & 0 & -1 \\ -1 & & & 1 & 0 \end{pmatrix}, \quad (3.3)$$

and solution

$$\mathbf{u}(t + \Delta t) = e^{\Delta t \mathbf{B}} \mathbf{u}(t). \quad (3.4)$$

The exact amplification factor ($\theta = k\Delta x$)

$$g_{ex} = e^{-i\eta \sin \theta}, \quad \text{with} \quad \eta = \frac{v\Delta t}{\Delta x}, \quad (3.5)$$

is unitary and causes a phase-shift of each Fourier component. In the limit of $\Delta x \rightarrow 0$, the phase-shift becomes uniform for all the Fourier components $e^{ikx} \rightarrow e^{ik(x-v\Delta t)}$, resulting in a uniform shift of the entire function $u(x) \rightarrow u(x-v\Delta t)$, which is the exact solution to (3.1). Any Taylor expansion of (3.4) will produce algorithms with a non-unitary g , resulting in unwanted dissipations or instability. The situation here is much more delicate than in the diffusion case.

The natural decomposition is similarly,

$$\mathbf{B} = \sum_{j=1}^N \mathbf{B}_j, \quad (3.6)$$

where

$$\mathbf{B}_j = \frac{v}{2\Delta x} \begin{pmatrix} \ddots & & & \\ & 0 & -1 & \\ & 1 & 0 & \\ & & & \ddots \end{pmatrix} \quad \text{with} \quad \mathbf{B}_N = \frac{v}{2\Delta x} \begin{pmatrix} 0 & & & 1 \\ & \ddots & & \\ & & \ddots & \\ -1 & & & 0 \end{pmatrix}. \quad (3.7)$$

It follows that

$$e^{\Delta t \mathbf{B}_j} = \begin{pmatrix} 1 & & & \\ c & -s & & \\ s & c & & \\ & & & 1 \end{pmatrix}, \quad e^{\Delta t \mathbf{B}_N} = \begin{pmatrix} c & & s \\ & 1 & \\ & & 1 \\ -s & & c \end{pmatrix}, \quad (3.8)$$

where now

$$c = \cos(\eta/2) \quad \text{and} \quad s = \sin(\eta/2). \quad (3.9)$$

Each $e^{\Delta t \mathbf{B}_j}$ only updates u_j and u_{j+1} as

$$\begin{aligned} u'_j &= cu_j - su_{j+1} \\ u'_{j+1} &= su_j + cu_{j+1}. \end{aligned} \quad (3.10)$$

As in the diffusion case, the above updating can be recasted into the following forms for algorithms 1A and 1B, with 1A given by

$$\begin{aligned} u_1^* &= cu_1 - su_2 \\ u'_2 &= su_1^* + u_2 - su_3 \\ u'_j &= su'_{j-1} + u_j - su_{j+1} \quad (2 < j < N) \\ u'_N &= su'_{N-1} + u_N - su_1^* \\ cu'_1 &= su'_N + cu_1 - su_2. \end{aligned} \quad (3.11)$$

and 1B given by

$$\begin{aligned} u_1^* &= su_N + cu_1 \\ u'_N &= su_{N-1} + u_N - su_1^* \\ u'_j &= su_{j-1} + u_j - su'_{j+1} \quad (2 < j < N) \\ u'_2 &= su_1^* + u_2 - su'_3 \\ cu'_1 &= su_N + cu_1 - su'_2. \end{aligned} \quad (3.12)$$

In contrast to the diffusion case, these algorithms are *not* exactly norm-preserving for periodic boundary condition. By adding up both sides of the above algorithms, one finds that what is preserved by 1A is not the usual norm $N = \sum_{j=1}^N u_j$, but a modified norm given by

$$\tilde{N}_{1A} = N + \left(\frac{c}{1-s} - 1\right)u_1 \quad (3.13)$$

Similarly, what is preserved by 1B is

$$\tilde{N}_{1B} = N + \left(\frac{c}{1+s} - 1\right)u_1. \quad (3.14)$$

If initially $u_1 = 0$, then $\tilde{N}_{1A} = \tilde{N}_{1B} = N_0$, where N_0 is the initial norm. As the system evolves, each algorithm's actual norm will evolve as

$$\begin{aligned} N_{1A} &= N_0 - \left(\frac{c}{1-s} - 1\right)u_1, \\ N_{1B} &= N_0 - \left(\frac{c}{1+s} - 1\right)u_1. \end{aligned} \quad (3.15)$$

The error is due to a single point u_1 , where it is the only point not updated twice immediately. As the wave form travels around the periodic box, u_1 will trace out the shape of the wave and imprint that as the error of the norm in time. For a sharp pulse, the norm error will return to zero after the pulse peak has passed through u_1 . Thus norm-preservation will be periodic. For the advection equation, this is a small effect, and is secondary to the phase and oscillation error mentioned below. However, this error will be important in the next section.

If the boundary values u'_1 , u'_2 and u'_N are ignored for now, then again the resulting second-order algorithm is unique, independent of the order of applying 1A or 1B. The amplification factors are all *unitary*:

$$g_{1A} = \frac{1 - se^{i\theta}}{1 - se^{-i\theta}} = \exp(-i\phi_{1A}), \quad (3.16)$$

$$g_{1B} = \frac{1 + se^{-i\theta}}{1 + se^{i\theta}} = \exp(-i\phi_{1B}), \quad (3.17)$$

$$\begin{aligned} g_2 &= g_{1B}(\Delta t/2)g_{1A}(\Delta t/2) \\ &= \frac{1 - i2(\tilde{s}/\tilde{c}^2)\sin\theta}{1 + i2(\tilde{s}/\tilde{c}^2)\sin\theta} = \exp(-i\phi_2), \end{aligned} \quad (3.18)$$

with phase angles

$$\begin{aligned}
\phi_{1A} &= 2 \tan^{-1} \left(\frac{s \sin \theta}{1 - s \cos \theta} \right), \\
\phi_{1B} &= 2 \tan^{-1} \left(\frac{s \sin \theta}{1 + s \cos \theta} \right), \\
\phi_2 &= \phi_{1A}(\Delta t/2) + \phi_{1B}(\Delta t/2), \\
&= 2 \tan^{-1} \left(\frac{2\tilde{s}}{1 - \tilde{s}^2} \sin \theta \right),
\end{aligned} \tag{3.19}$$

where here

$$\tilde{s} = \sin(\eta/4) \quad \text{and} \quad \tilde{c} = \cos(\eta/4). \tag{3.20}$$

Since g_{1A} and g_{1B} are not complex conjugate of each other, their phase errors do not exactly cancel. Their residual difference is the error of the second-order algorithm.

Algorithms (3.11) and (3.12) are the corresponding Saul'yev's schemes for solving the advection equation. The coefficient here is $s = \sin(\eta/2)$ rather than Saul'yev's coefficient of $s = \eta/2$. This explains why it makes no sense to apply Saul'yev's schemes at $s > 1$, since they can no longer be derived from the fundamental updating matrix (3.10) with a real $c = \sqrt{1 - s^2}$. At $s > 1$, Saul'yev's schemes are in fact unstable, suffering from spatial amplification²⁴, despite the unimodulus appearance of (3.16) and (3.17). This is easy to see in the case of algorithm 1A. If initially $u_j = 0$ for $j \geq J$, but $u_{J-1} \neq 0$, then according to (3.11), $u'_{J+n} = s^{n+1} u'_{J-1}$ increases without bound as a function of n . Even the case of $s = 1$ is pathological. For Saul'yev's coefficient $s = \eta/2 = 1$, one has

$$g_{1A} = -e^{ik\Delta x} = -e^{ik(v/2)\Delta t} \quad \text{and} \quad g_{1B} = e^{-ik\Delta x} = e^{-ik(v/2)\Delta t}. \tag{3.21}$$

Under algorithm 1A, Fourier mode e^{ikx} will flip its sign and propagate with velocity $-v/2$. Under 1B, it will propagate with velocity $v/2$. The resulting second order algorithm then leaves the Fourier mode *stationary* with only a sign flip. This is completely contrary to the behavior of the exact solution and is a source of great error for Saul'yev's schemes. As we will show below, alternative choices for s will eliminate such unphysical behaviors.

While the derived choice of $s = \sin(\eta/2)$ is unconditionally stable for all η , the resulting algorithms 1A and 1B have *huge* phase errors, and are no better than Saul'yev's choice of $s = \eta/2$. This is because in comparison with the exact phase angle,

$$\phi_{ex} = \eta \sin \theta = \eta \theta - \frac{\eta}{6} \theta^3 + \dots \tag{3.22}$$

algorithms 1A and 1B have expansions

$$\begin{aligned}
\phi_{1A} &= \frac{2s}{1-s} \theta - \frac{s(1+s)}{3(1-s)^3} \theta^3 + \dots, \\
\phi_{1B} &= \frac{2s}{1+s} \theta - \frac{s(1-s)}{3(1+s)^3} \theta^3 + \dots,
\end{aligned} \tag{3.23}$$

and neither $s = \eta/2$ nor $s = \sin(\eta/2)$ can result in a first-order coefficient of θ matching that of ϕ_{ex} exactly. The choices of s that can do this are, for 1A,

$$\frac{2s}{1-s} = \eta \quad \rightarrow \quad s = \frac{\eta}{2+\eta}, \tag{3.24}$$

and for 1B,

$$\frac{2s}{1+s} = \eta \quad \rightarrow \quad s = \frac{\eta}{2-\eta}. \tag{3.25}$$

This then reproduces the Roberts and Weiss^{23,24} forms of the Saul'yev-type algorithm and will be denoted as RW1A and RW1B. For $\eta > 0$, only RW1A is unconditionally stable and RW1B is limited by spatial amplification to $\eta < 1$. The pathological behavior of 1A at $s = 1$ can no longer occur at any finite η .

For the above choices of s , the corresponding phase angles are

$$\begin{aligned}\phi_{1A} &= \eta\theta - \left(\frac{\eta}{6} + \frac{\eta^2}{4} + \frac{\eta^3}{12}\right)\theta^3 + \dots, \\ \phi_{1B} &= \eta\theta - \left(\frac{\eta}{6} - \frac{\eta^2}{4} + \frac{\eta^3}{12}\right)\theta^3 + \dots,\end{aligned}\tag{3.26}$$

and the modified norms are

$$\tilde{N}_{1A} = N + (\sqrt{1+\eta} - 1)u_1,\tag{3.27}$$

$$\tilde{N}_{1B} = N + (\sqrt{1-\eta} - 1)u_1.\tag{3.28}$$

The second order algorithm from concatenating RW1A and RW1B is

$$\begin{aligned}\phi_2 &= \phi_{1A}(\eta/2) + \phi_{1B}(\eta/2) \\ &= \eta\theta - \left(\frac{\eta}{6} + \frac{\eta^3}{48}\right)\theta^3 + \left(\frac{\eta}{120} + \frac{5\eta^3}{192} + \frac{\eta^5}{1280}\right)\theta^5 + \dots\end{aligned}\tag{3.29}$$

This second-order advection algorithm will be denoted as RW2. Because RW1B is limited by spatial amplification to $\eta < 1$, RW2 is limited in stability to $\eta < 2$.

To generate a stable second-order algorithm for all η , one can concatenate 1A and 1B with the same \tilde{s} . To match g_{ex} to first order in θ then requires

$$\frac{2\tilde{s}}{1-\tilde{s}} + \frac{2\tilde{s}}{1+\tilde{s}} = \eta \quad \rightarrow \quad \frac{2\tilde{s}}{1-\tilde{s}^2} = \frac{\eta}{2} \quad \rightarrow \quad \tilde{s} = \frac{2}{\eta} \left(\sqrt{1 + \frac{\eta^2}{4}} - 1 \right).\tag{3.30}$$

The resulting amplification factor is, according to (3.18),

$$g_2 = \frac{1 - i(\eta/2) \sin \theta}{1 + i(\eta/2) \sin \theta},\tag{3.31}$$

which is precisely the *implicit* Crank-Nicolson amplification factor. Since by (3.30), $\tilde{s} \leq 1$, and $\tilde{c} = \sqrt{1 - \tilde{s}^2}$ is well-defined for all η , the algorithm is unconditionally stable and can be applied to periodic boundary problems via the the fundamental updating (3.10). Corresponding to (3.31), the phase-angle has the characteristic expansion,

$$\phi_2 = \eta\theta - \left(\frac{\eta}{6} + \frac{\eta^3}{12}\right)\theta^3 + \left(\frac{\eta}{120} + \frac{\eta^3}{24} + \frac{3\eta^5}{240}\right)\theta^5 + \dots\tag{3.32}$$

$$= \eta \sin \theta + \eta^3 F_3(\theta) + \eta^5 F_5(\theta) + \dots\tag{3.33}$$

where now the time parameter is η and the original ‘‘Hamiltonian’’ is $h_0 = \sin \theta$. We shall designate this second-order algorithm, with \tilde{s} given by (3.30), as A2C. The second-order algorithm corresponding to Saul’pev’s choice of $\tilde{s} = \eta/4$ will be denoted as A2S, and the initially derived result of $\tilde{s} = \sin(\eta/4)$ as A2.

On the left of Fig.4, the phase error of these symplectic algorithms are exaggerated and compared to the explicit but dissipative Lax-Wendroff (LW) scheme at a large value of $\eta = 0.7$. The original 1A and 1B algorithms have huge phase errors but are mostly cancelled in the second-order algorithm A2. Even so, algorithm A2’s error curve has a finite slope at $\theta = 0$, as shown on the right of Fig.4. By construction, schemes RW1A, RW1B, RW2, A2C and LW all have zero error slopes at $\theta = 0$. This is a crucial advantage of A2C over A2. Also, A2C is stable for all η , while RW2 is limited by spatial amplification to $\eta < 2$.

The importance of having a zero error slope in the phase angle can be better appreciated from the following considerations. Let

$$\chi(t) = \frac{\int x u(x, t) dx}{\int u(x, t) dx}.\tag{3.34}$$

Since the solution to the advection equation is $u(x, t) = u_0(x - vt)$, we have the exact result

$$\chi(\Delta t) = \frac{\int x u_0(x - v\Delta t) dx}{\int u_0(x - v\Delta t) dx} = \frac{\int (y + v\Delta t) u_0(y) dy}{\int u_0(y) dy} = \chi(0) + v\Delta t.\tag{3.35}$$

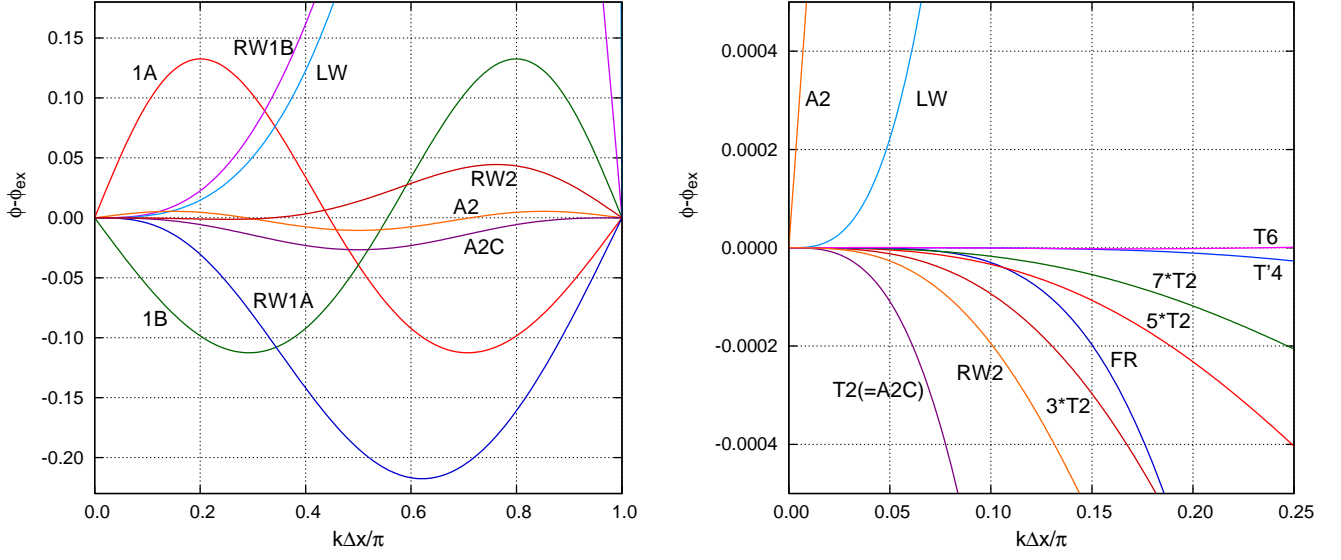


FIG. 4: The phase error of various advection algorithms at $\eta = 0.7$. **Left:** The unmodified symplectic algorithms are denoted as 1A, 1B, and A2. The Robert-Weiss versions are denoted as RW1A, RW1B and RW2. A2C is the second-order algorithm with the Crank-Nicolson amplification factor. LW is the Lax-Wendroff scheme included for comparison. **Right:** The phase errors of fourth and sixth-order algorithms composed out of three, five and seven second-order algorithms T_2 . Their phase errors are compared to that of running the second-order algorithm three, five and seven times at reduced time steps. The T_2 used here is A2C. Previous second-order algorithms are also included for a close-up comparison.

Multiplying algorithm 1A (3.11) by j and sum over j yields

$$\begin{aligned} \sum_{j=1}^N j u'_j - s \sum_{j=1}^N j u'_{j-1} &= \sum_{j=1}^N j u_j - s \sum_{j=1}^N j u_{j+1} \\ (1-s) \sum_{j=1}^N j u'_j - s \sum_{j=1}^N u'_{j-1} &= (1-s) \sum_{j=1}^N j u_j + s \sum_{j=1}^N u_{j+1}. \end{aligned} \quad (3.36)$$

For a localized pulse far from the boundary, the norm can be consider conserved,

$$\sum_{j=1}^N u'_{j-1} = \sum_{j=1}^N u_{j+1}. \quad (3.37)$$

One then has the discrete version of (3.35)

$$\frac{\sum_{j=1}^N (j \Delta x) u'_j}{\sum_{j=1}^N u'_j} = \frac{\sum_{j=1}^N (j \Delta x) u_j}{\sum_{j=1}^N u_j} + \frac{2s}{1-s} \Delta x \quad (3.38)$$

which will reproduce the displacement exactly if

$$\frac{2s}{1-s} = \frac{v \Delta t}{\Delta x} = \eta, \quad (3.39)$$

which is the condition (3.24) for a zero error-slope. Similarly for 1B satisfying (3.25). Far from the boundary, symplectic advection algorithms with a zero-error slope in the phase angle would exactly preserve the first two moments of $\langle x^n \rangle$.

In Fig.5 we show the working of these algorithms in propagating an initial profile

$$u(x, 0) = \exp \left[- \left(\frac{x}{2} \right)^6 \right]. \quad (3.40)$$

The power of 6 was chosen to provide a steep, but continuous profile so that both the phase error and the oscillation error are visible. If the profile were too steep, like that of a square wave, the oscillation error would have overwhelmed the calculation before the phase error can be seen. The oscillation errors in all these symplectic algorithms are primarily due to the oscillation error in algorithm 1A. Algorithm 1B has a much smaller oscillation error. Because all the algorithms are essentially norm-preserving, oscillation errors is inherent to any scheme which does not preserve the positivity of the solution²¹.

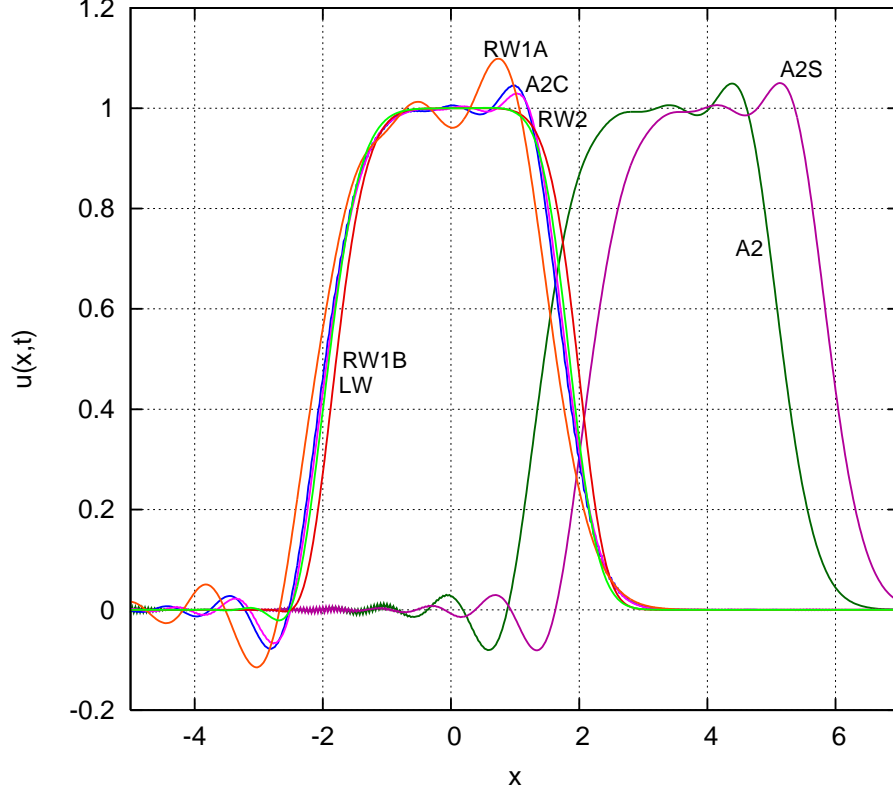


FIG. 5: The propagation of initial profile (3.40) fives times around a periodic box of $[-10,10]$ with $\Delta x = 0.025$, $\Delta t = 0.02$, $v = 1$, and $\eta = 0.8$, corresponding to 5000 iterations of each algorithm. If there were no phase error, the profile would remain centered on $x = 0$. Algorithms A2 and A2S have large and positive, phase errors. The oscillation errors in these second-order symplectic algorithms are predominately due to the imbedded 1A algorithm. Algorithm RW1B has a much smaller oscillation error and is comparable to the oscillation error in the dissipative Lax-Wendroff scheme (bright green line).

Higher order advection algorithms can again be constructed by the method of composition (1.17). Since g_2 is unitary, any product of g_2 is also unitary. Thus to preserve unitarity, one must use only a single product composition, rather than a multi-product expansion as in the diffusion case. (However, as noted in the last section, this violation of unitarity in MPE is small with increasing order. At sufficiently high order, this violation is beyond machine precision and is indistinguishable from a truly unitary algorithm²⁰. For simplicity, we will only consider strictly unitary algorithms in this discussion.) For a single product composition, the resulting phase angle is just a sum of ϕ_2 's. The simplest fourth-order composition, the Forest-Ruth (FR) algorithm⁷⁻⁹ is given by

$$T_4^{FR}(\Delta t) = T_2(a_1 \Delta t) T_2(a_0 \Delta t) T_2(a_1 \Delta t), \quad (3.41)$$

with $a_1 = 1/(2 - b)$, $a_0 = -b/(2 - b)$ and $b = 2^{1/3}$. The coefficients a_0 and a_1 satisfy the consistency condition $2a_1 + a_0 = 1$ and the fourth-order condition $2a_1^3 + a_0^3 = 0$. If we take $T_2(\Delta t)$ to be A2C, then the phase angle for $T_4^{FR}(\Delta t)$ is just (3.32) with all η^3 terms removed,

$$\phi_4^{FR} = \eta\theta - \frac{\eta}{6}\theta^3 + \left(\frac{\eta}{120} - 5.29145 \frac{3\eta^5}{240} \right) \theta^5 + \dots, \quad (3.42)$$

which is then correct to fourth-order in θ . This is not true if we take $T_2(\Delta t)$ to be the original algorithm A2. That fourth-order time-marching algorithm's phase angle will still have a small error slope at $\theta = 0$. One should therefore only uses A2C to compose higher order algorithms.

The large numerical coefficient in (3.42) is due to $2a_1^5 + a_0^5 = -5.29145$, reflecting the fact that FR has a rather large residual error. A better fourth-order algorithm advocated by Suzuki¹⁰ (S4) at the expense of two more T_2 is

$$T_4^S = T_2(a_1 \Delta t) T_2(a_1 \Delta t) T_2(a_0 \Delta t) T_2(a_1 \Delta t) T_2(a_1 \Delta t) \quad (3.43)$$

where now $a_1 = 1/(4 - 4^{1/3})$, $a_0 = -4^{1/3} a_1$, and $4a_1^5 + a_0^5 = -0.074376$, which is nearly sixth-order.

At the expense of two more T_2 , one can achieve sixth-order via Yoshida's algorithm⁹ (Y6),

$$T_6^Y = T_2(a_3 \Delta t) T_2(a_2 \Delta t) T_2(a_1 \Delta t) T_2(a_0 \Delta t) T_2(a_1 \Delta t) T_2(a_2 \Delta t) T_2(a_3 \Delta t) \quad (3.44)$$

with coefficients

$$\begin{aligned} a_1 &= -1.17767998417887 & a_2 &= 0.235573213359357 \\ a_3 &= 0.784513610477560 & \text{and } a_0 &= 1 - 2(c_1 + c_2 + c_3). \end{aligned} \quad (3.45)$$

For eighth and higher order algorithms, see Refs.12 and 25.

The phase errors of these higher order algorithms at $\eta = 0.7$ are shown at the right of Fig.4. Since each algorithm applies T_2 (taken to be A2C) n ($=3,5,7$) times, they are compared to the phase error of running T_2 n times at a reduce time step of $\Delta t/n$. Algorithms S4 and Y6 beat their target comparisons by orders of magnitude. There is a clear advantage in going to higher-order algorithms for solving the advection equation.

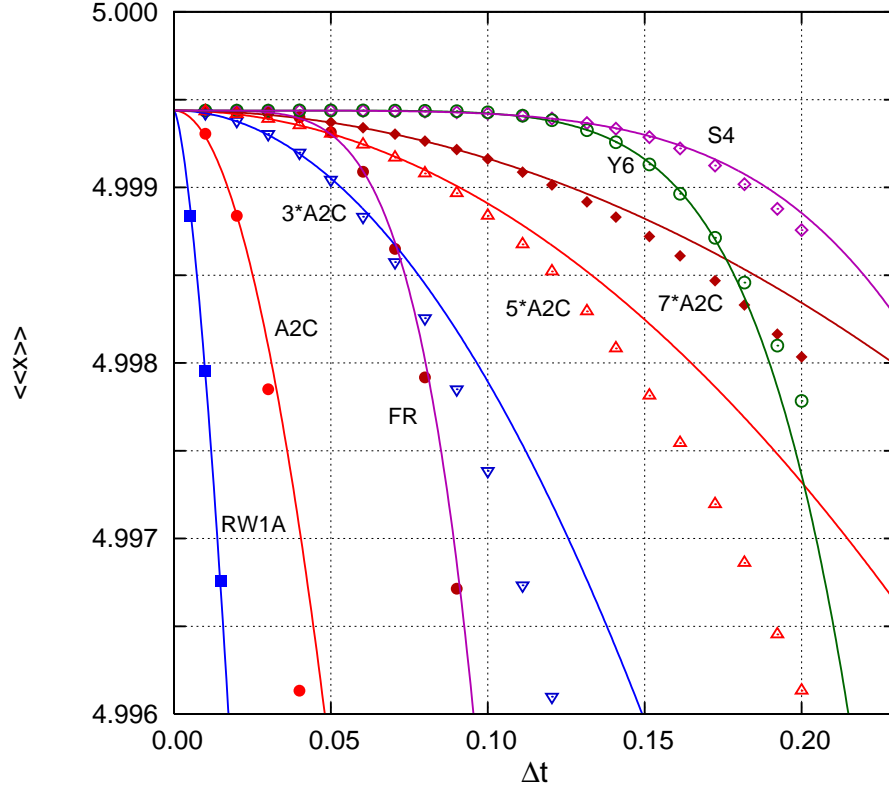


FIG. 6: Comparing the convergence of various unconditionally stable symplectic advection algorithms. The solid lines are power laws of the form $a + b\Delta t^n$ seeking to verify the order of the algorithm. Higher order algorithms composed of n second-order algorithms A2C are compared to n^* A2C at a reduce time-step size of $\Delta t/n$. FR, S4 and Y6 requires 3, 5 and 7 runs of A2C respectively. In this computation of (3.46), all algorithms met or exceeded their nominal order of convergence. See text for details.

In Fig.6, the convergence of these higher order algorithms are compared. The range of the time steps used, $\Delta t = 0.01 - 0.20$, corresponds to $\eta = 0.4 - 8.0$. The same profile (3.40) is initially centered at $x = -5$ and propagated to $x = 5$ at $v = 1$. The time steps are chosen as $\Delta t = 10/m$, so that m iterations exactly give $t = 10$. Since all algorithms satisfy (3.35) despite the oscillation errors, we compute the expectation value

$$\langle\langle x \rangle\rangle = \frac{\sum_{j=1}^N (j\Delta x) |u_j|}{\sum_{j=1}^N |u_j|} \quad (3.46)$$

with respect to the absolute value of the propagated profile. The phase errors for A2 and A2S are known to be large from Fig.5, and are not included in this comparison. The solid lines are fitted power laws of the form $a + b\Delta t^n$. The first surprise is that RW1A's result cannot be fitted with $n = 1$. The fitted line is a fit with $n = 3/2$. The algorithm A2C can be well fitted with $n = 2$. This is specially clear in the case where A2C is applied seven times at step size $\Delta t/7$. The fourth-order Forest-Ruth (FR) and Suzuki (S4) algorithms can only be fitted with $n = 5$, and the sixth-order Yoshida (Y6) algorithm with $n = 7$. They all converged to a value of $a = 4.999438$ which is below the exact value of 5. This is related to the grid size error. Halving the grid size to $\Delta x = 0.0125$ gives $a = 4.999997$.

IV. SYMPLECTIC ADVECTION-DIFFUSION ALGORITHMS

The advection-diffusion equation,

$$\frac{\partial u}{\partial t} = -v \frac{\partial u}{\partial x} + D \frac{\partial^2 u}{\partial x^2}, \quad (4.1)$$

has the exact operator solution

$$u(x, \Delta t) = e^{-v\Delta t \frac{\partial}{\partial x} + D\Delta t \frac{\partial^2}{\partial x^2}} u(x, 0). \quad (4.2)$$

If v and D are just constants, then since $[\frac{\partial}{\partial x}, \frac{\partial^2}{\partial x^2}] = 0$, one has

$$\begin{aligned} u(x, \Delta t) &= e^{-v\Delta t \frac{\partial}{\partial x}} e^{D\Delta t \frac{\partial^2}{\partial x^2}} u(x, 0) \\ &= e^{-v\Delta t \frac{\partial}{\partial x}} \tilde{u}(x, \Delta t) \\ &= \tilde{u}(x - v\Delta t, \Delta t), \end{aligned} \quad (4.3)$$

where $\tilde{u}(x, \Delta t)$ is the diffused solution. The complete solution is therefore the exact diffused solution $\tilde{u}(x, \Delta t)$ displaced by $v\Delta t$.

For periodic boundary condition, our matrices *also* commute, $[\mathbf{A}, \mathbf{B}] = 0$, so that the discretized version also holds,

$$\mathbf{u}(t + \Delta t) = e^{\Delta t \mathbf{A}} e^{\Delta t \mathbf{B}} \mathbf{u}(t). \quad (4.4)$$

Thus arbitrary high order algorithms can be obtained by applying higher order advection and diffusion algorithms in turns from the previous sections. In the case of spatially dependent $D(x)$ or $v(x)$ where $[\mathbf{A}, \mathbf{B}] \neq 0$, one can do the second-order splitting

$$\mathbf{u}(t + \Delta t) = e^{\frac{1}{2}\Delta t \mathbf{A}} e^{\Delta t \mathbf{B}} e^{\frac{1}{2}\Delta t \mathbf{A}} \mathbf{u}(t) \quad (4.5)$$

and apply higher order MPE algorithms. However, for periodic boundary condition, this way of solving the advection-diffusion equation cannot conserve the norm. Consider the case of applying the advection algorithms RW1A, RW1B followed by any norm-conserving diffusion algorithm. From (3.27), the change in the modified norm after Δt would be

$$\tilde{N}'_{1A} - \tilde{N}_{1A} = (\sqrt{1 + \eta} - 1)(u'_1 - u_1). \quad (4.6)$$

For RW1A, u_1 is a discontinuous point *higher* than its adjacent neighbors u_n and u_2 . Consequently, after the diffusion step, $u'_1 < u_1$ and there is a loss of normalization in (4.6). For RW1B, u_1 is a discontinuous point *lower* than its adjacent neighbors u_n and u_2 . After the diffusion step, $u'_1 > u_1$, and again results in a loss of normalization:

$$\tilde{N}'_{1B} - \tilde{N}_{1B} = (\sqrt{1 - \eta} - 1)(u'_1 - u_1). \quad (4.7)$$

As will be shown, this loss is small for small D , but is irreversible and accumulative after each orbit around the periodic box. For fixed boundary with $u_1 = 0$, there is no such norm-conserving problem.

An alternative is to update the advection and diffusion steps simultaneously. In this case, one might decomposing $\mathbf{A} + \mathbf{B}$ into a sum of 2×2 matrices as done previously,

$$\mathbf{C}_j = \mathbf{A}_j + \mathbf{B}_j = \frac{D}{\Delta x^2} \begin{pmatrix} \ddots & & & \\ & -1 & 1 & \\ & 1 & -1 & \\ & & & \ddots \end{pmatrix} + \frac{v}{2\Delta x} \begin{pmatrix} \ddots & & & \\ & 0 & -1 & \\ & 1 & 0 & \\ & & & \ddots \end{pmatrix} \quad (4.8)$$

resulting in

$$e^{\Delta t \mathbf{C}_j} = \begin{pmatrix} 1 & & & \\ & \alpha & \lambda & \\ & \beta & \alpha & \\ & & & 1 \end{pmatrix}, \quad e^{\Delta t \mathbf{C}_N} = \begin{pmatrix} \alpha & & & \lambda \\ & 1 & & \\ & & 1 & \\ \beta & & & \alpha \end{pmatrix}, \quad (4.9)$$

with

$$\alpha = e^{-r} \cosh \psi, \quad \beta = e^{-r} (r + \eta/2) \frac{\sinh \psi}{\psi}, \quad \lambda = e^{-r} (r - \eta/2) \frac{\sinh \psi}{\psi}, \quad (4.10)$$

and $\psi = \sqrt{r^2 - (\eta/2)^2}$. The corresponding Saul'pev form of the 1A algorithm is then

$$u'_j = \beta u'_{j-1} + \gamma u_j + \lambda u_{j+1} \quad (4.11)$$

where $\gamma = \alpha^2 - \beta\lambda = e^{-2r}$ remains the determinant of the updating matrix. However, for the Saul'pev form (4.11) to be norm-preserving, one must have

$$\beta + \gamma + \lambda = 1. \quad (4.12)$$

Surprisingly, this is grossly violated by (4.10) when both r and η are non-vanishing.

As we have learned in the previous two sections, any such initial algorithm can be far from optimal. Therefore, one may as well begin with an assumed updating matrix,

$$\begin{aligned} u'_j &= \alpha u_j + \lambda u_{j+1} \\ u'_{j+1} &= \beta u_j + \alpha u_{j+1} \end{aligned} \quad (4.13)$$

and determine its elements by enforcing norm-conserving condition (4.12) and by matching the expansion coefficients of the exact amplification factor. The sequential applications of this updating matrix yields Saul'pev-type algorithms 1A (4.11) and 1B,

$$u'_j = \beta u_{j-1} + \gamma u_j + \lambda u'_{j+1} \quad (4.14)$$

The determinant $\gamma = \alpha^2 - \beta\lambda$ is to be regarded as fixing α as a function of γ and β via $\alpha = \sqrt{\gamma + \beta\lambda}$. The resulting amplification factors are then

$$\begin{aligned} g_{1A} &= \frac{\gamma + \lambda e^{i\theta}}{1 - \beta e^{-i\theta}} = e^{-h_{1A}}, \\ g_{1B} &= \frac{\gamma + \beta e^{-i\theta}}{1 - \lambda e^{i\theta}} = e^{-h_{1B}}. \end{aligned} \quad (4.15)$$

The norm condition (4.12) fixes λ in terms of β and γ . In terms of γ and β algorithms 1A and 1B have expansions,

$$\begin{aligned} h_{1A} &= \frac{2\beta - (1 - \gamma)}{1 - \beta} i\theta + \frac{(1 - \gamma)(\beta + \gamma)}{2(1 - \beta)^2} \theta^2 + O(\theta^3) \\ h_{1B} &= \frac{2\beta - (1 - \gamma)}{\gamma + \beta} i\theta + \frac{(1 - \gamma)(1 - \beta)}{2(\gamma + \beta)^2} \theta^2 + O(\theta^3) \end{aligned} \quad (4.16)$$

Matching the first and second order coefficients of the exact exponent

$$\begin{aligned} h_{ex} &= i\eta \sin(\theta) + 4r \sin(\theta/2)^2 \\ &= i\eta\theta + r\theta^2 + O(\theta^3), \end{aligned} \quad (4.17)$$

then completely determines, for 1A and 1B respectively,

$$\beta = \frac{1 - \gamma + \eta}{2 + \eta} \quad \gamma = \frac{1 - wr}{1 + wr} \quad w = \frac{2}{2 + \eta(3 + \eta)}, \quad (4.18)$$

$$\beta = \frac{1 - \gamma + \gamma\eta}{2 - \eta} \quad \gamma = \frac{1 - wr}{1 + wr} \quad w = \frac{2}{2 - \eta(3 - \eta)}. \quad (4.19)$$

These are the generalized Roberts-Weiss algorithms for the advection-diffusion equation.

For any choice of β and γ , the modified norm including the boundary effect now reads

$$\tilde{N}_{1A} = N + \left(\frac{\alpha}{1-\beta} - 1\right)u_1 \quad (4.20)$$

$$\tilde{N}_{1B} = N + \left(\frac{\alpha}{1-\lambda} - 1\right)u_1 \quad (4.21)$$

Remarkably, for the above generalized RW algorithms, one has

$$\frac{\alpha}{1-\beta} = \sqrt{1+\eta} \quad \text{and} \quad \frac{\alpha}{1-\lambda} = \sqrt{1-\eta}. \quad (4.22)$$

The modified norms (4.20) and (4.21) are therefore the same as the pure advection cases of (3.27) and (3.28). For these two first order advection-diffusion algorithms, their norm-conservation in a periodic box will then be periodic, as in the pure advection case. This is shown in Fig.7. However, as soon as one concatenate them into algorithm RW2, the loss of norm is irreversible. This is because each algorithm will behave as a diffusion algorithm for the other. The norm-loss mechanism described earlier will then apply.

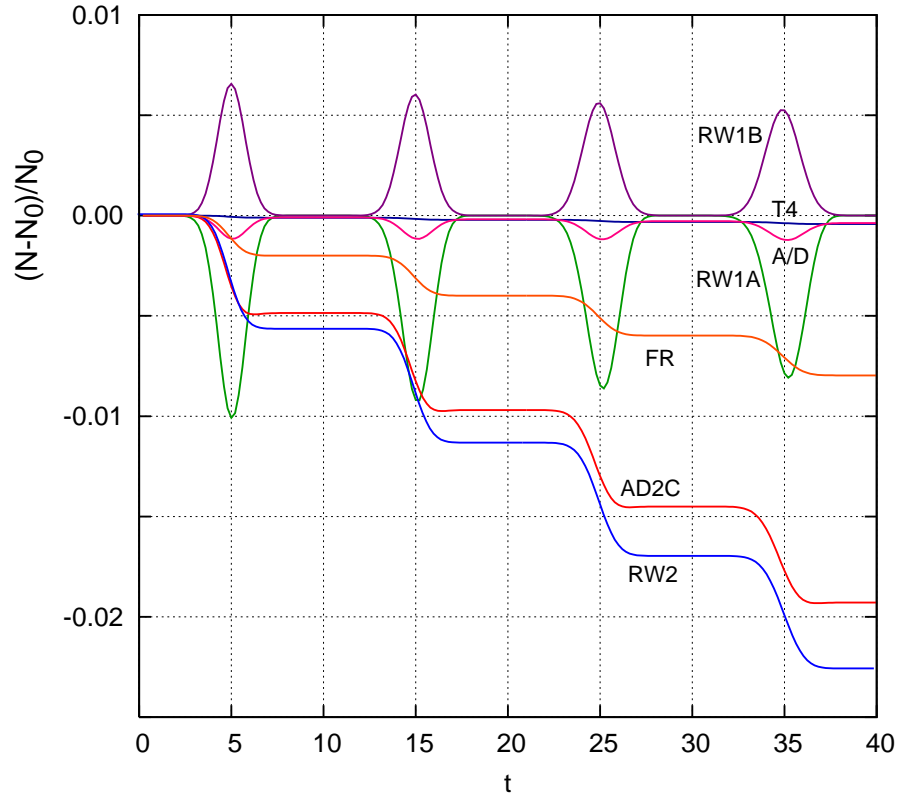


FIG. 7: The normalization error of various algorithms when propagating a Gaussian profile in a periodic box of $[0,10]$ with $\Delta x = 0.05$, $\Delta t = 0.033$, $v = 1$, $D = 0.005$, $\eta = 0.66$ and $r = 0.066$. The profile is initially centered at $x = 5$. At $t = 5, 15, 25, 35$, the Gaussian peak is at the edge of the periodic box. First-order advection-diffusion algorithms RW1A and RW1B conserve the norm periodically. All higher than first-order algorithms suffer loss of normalization irreversibly, though very small for MPE algorithm T4 and A/D. The latter is applying the advection algorithm A2C and the diffusion algorithm D2S sequentially.

The second-order algorithm's amplification factor is

$$g_2 = \left(\frac{\tilde{\gamma} + \tilde{\lambda}e^{i\theta}}{1 - \tilde{\beta}e^{-i\theta}} \right) \left(\frac{\tilde{\gamma} + \tilde{\beta}e^{-i\theta}}{1 - \tilde{\lambda}e^{+i\theta}} \right) = e^{-h_2}, \quad (4.23)$$

where $\tilde{\gamma} = \gamma(\Delta t/2)$, etc.. In terms of $\tilde{\gamma}$ and $\tilde{\beta}$, h_2 has the expansion,

$$h_2 = i\theta \left(\frac{(1 + \tilde{\gamma})(\tilde{\gamma} - 1 + 2\tilde{\beta})}{(1 - \tilde{\beta})(\tilde{\gamma} + \tilde{\beta})} \right) + O(\theta^2). \quad (4.24)$$

Matching this to the first order coefficient of the exact exponent (4.17) determines

$$\tilde{\beta} = \frac{1}{2}(1 - \tilde{\gamma}) + \frac{1}{2}(1 + \tilde{\gamma})\tilde{s}, \quad (4.25)$$

and

$$\tilde{\lambda} = \frac{1}{2}(1 - \tilde{\gamma}) - \frac{1}{2}(1 + \tilde{\gamma})\tilde{s}. \quad (4.26)$$

where \tilde{s} has been previously defined by (3.30). In terms of only $\tilde{\gamma}$,

$$h_2 = i\eta\theta + 2\frac{(1 - \tilde{\gamma})}{(1 + \tilde{\gamma})}\frac{(1 + 3\tilde{s}^2)}{(1 - \tilde{s}^2)^2}\theta^2 + O(\theta^3), \quad (4.27)$$

and matching the second order coefficient in (4.17) determines

$$\tilde{\gamma} = \frac{1 - wr/2}{1 + wr/2} \quad \text{with} \quad w = (1 - \tilde{s}^2)^2/(1 + 3\tilde{s}^2). \quad (4.28)$$

If $\eta = 0$, $\tilde{s} = 0$, one recovers (2.33), which is the second-order diffusion algorithm D2S. If $r = 0$, then $\gamma = 1$ and one recovers the second-order advection algorithm A2C with $\tilde{\lambda} = -\tilde{\beta} = -\tilde{s}$. We shall refer to this second-order algorithm as AD2C. AD2C is an unconditionally stable algorithm which requires only half the effort of algorithm A/D, which applies A2C and D2S sequentially. However, AD2C has a greater irreversible norm-error when applied to periodic boundary problems. This is shown in Fig.7.

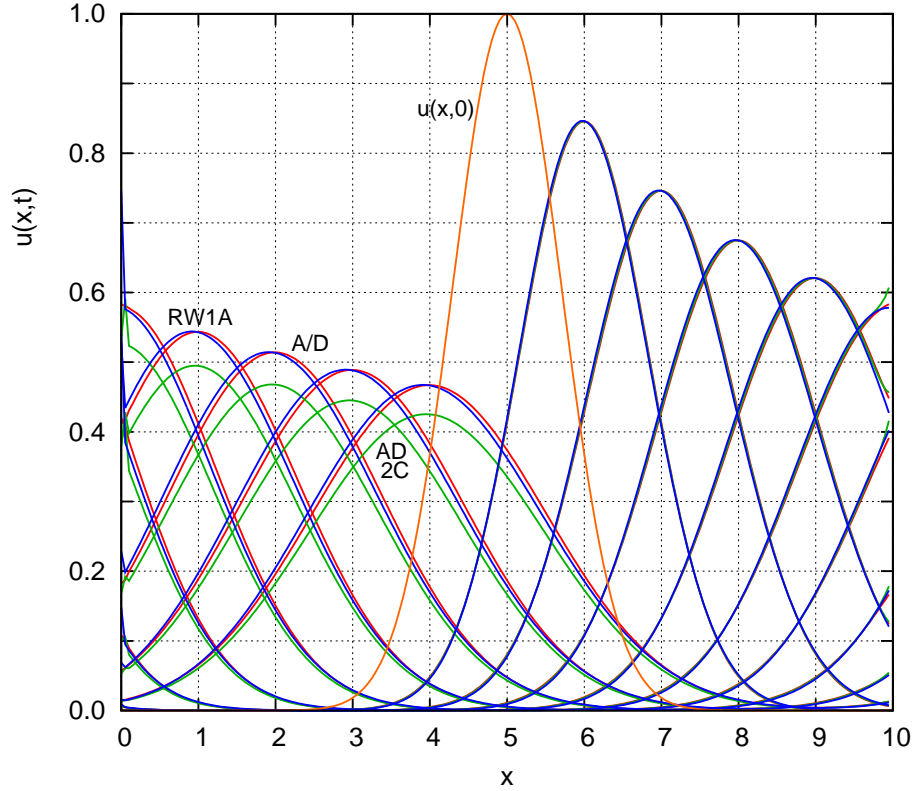


FIG. 8: The propagation of a Gaussian profile in a periodic box of $[0,10]$ with $\Delta x = 0.05$, $\Delta t = 0.033$, $v = 1$, $D = 0.1$, $\eta = 0.66$ and $r = 1.33$. The profile is initially centered at $x = 5$. All three profiles produced by algorithms RW1A, AD2C and A/D are in essential agreement prior to the pulse peak hitting the right periodic edge. As the profiles reappear from the left, the norm of AD2C is noticeably lower.

This norm-error for periodic boundary condition can be greatly reduced by going to higher orders. Fig.7 shows the result for the fourth-order MPE algorithm T_4 , with AD2C as T_2 . For r small, surprisingly, even the negative-coefficient algorithm FR is stable, but with an error comparable to second-order algorithms.

In Fig.8 we illustrate the effect of this norm-loss error at a large value of $r = 1.33$. For clarity, only results from three representative algorithms are shown. Algorithms A/D and RW1A have small or only periodic norm-losses and remained in agreement after the Gaussian peak has reappeared from the left. However, algorithm AD2C suffers an irreversible norm-loss and its peak is noticeably lower.

V. CONCLUDING SUMMARY

In this work, we have shown that explicit symplectic finite-difference methods can be derived in the same way as symplectic integrators by exponential splittings. The resulting sequential updating algorithms reproduce Saul'pev's unconditionally stable schemes, but is more general and can be applied to periodic boundary problems. In contrast to Saul'pev's original approach, where the algorithm is fixed by its derivation, symplectic algorithms can be systematically improved by matching the algorithm's amplification factor more closely to the amplification factor of the semi-discretized equation. One key contribution of this work is the recognition that, for finite difference schemes, their amplification factors should be compared, not to the continuum growth factor, but to the amplification factor of the semi-discretized equation. The exponent of this amplification factor then serve as the "Hamiltonian" for developing symplectic finite-difference algorithms. By requiring the algorithm's modified "Hamiltonian" to match the original "Hamiltonian" to the leading order, one produces all known, non-pathological first-order Saul'pev schemes and many new second-order algorithms for solving the diffusion and the advection equation. As a consequence of this formal correspondence with symplectic integrators, existing methods of generating higher order integrators can be immediately used to produce higher order finite-difference schemes.

The generalization to higher dimensions can be done by dimensional splitting, resulting in unconditionally stable, alternate-direction-explicit methods. The generalization to non-constant diffusion and advection coefficients is a topic suitable for a future study. The coefficients must frozen in such a way that one can recover Saul'pev's asymmetric schemes from their more basic sequential updateings.

Acknowledgments

This work is supported in part by the Austrian FWF grant P21924 and the Qatar National Research Fund (QNRF) National Priority Research Project (NPRP) grant # 5-674-1-114. I thank my colleague Eckhard Krotscheck and the Institute for Theoretical Physics at the Johannes Kepler University, Linz, Austria, for their wonderful hospitality during the summers of 2010-2012.

-
- ¹ R. Courant, K.O. Friedrichs, H. Lewy "Über die partiellen Differenzengleichungen der mathematischen Physik", *Maht. Anal.* **100**, 32-74 (1928).
 - ² V. K. Saul'yev, "On a method of numerical integration of a diffusion equation", *Dokl. Akad. Nauk. SSSR*, (in Russian) **115**, 1077-1079 (1957).
 - ³ V. K. Saul'yev, *Integration of equation of parabolic type by the method of nets*, Pergamon Press, New York, 1964.
 - ⁴ B. K. Larkin, "Some stable explicit difference approximation to the diffusion equation", *Math. Comput.* **18**, 196-201 (1964).
 - ⁵ D. J. Evans and A. R. B. Abdullah, "Group Explicit Methods for Parabolic Equations", *Intern. J. Computer Math.* **14**, 73-105 (1983).
 - ⁶ D. J. Evans, "Alternating group explicit methods the diffusion equations", *App. Math. Modelling*, **9**, 201-206 (1985).
 - ⁷ M. Creutz and A. Gocksch, "Higher-order hybrid Monte-Carlo algorithms", *Phys. Rev. Letts.* **63**, 9 (1989).
 - ⁸ E. Forest and R. D. Ruth, "4th-order symplectic integration", *Physica D* **43**, 105 (1990).
 - ⁹ H. Yoshida, "Construction of higher order symplectic integrators", *Phys. Lett.* **A150**, 262-268, (1990).
 - ¹⁰ M. Suzuki, "Hybrid exponential product formulas for unbounded operators with possible applications to Monte Carlo simulations", *Phys. Lett.* **A 146**, 319 (1990).
 - ¹¹ H. Yoshida, "Recent progress in the theory and application of symplectic integrators", *Celest. Mech. Dyn. Astron.* **56**, 27 (1993).
 - ¹² E. Hairer, C. Lubich, and G. Wanner, *Geometric Numerical Integration*, Springer-Verlag, Berlin-New York, 2002.
 - ¹³ R. I. McLachlan and G. R. W. Quispel, "Splitting methods", *Acta Numerica* **11**, 241 (2002).
 - ¹⁴ H. F. Trotter, "Approximation of semi-groups of operators", *Pacific J. Math.* **8**, 887-919 (1958).
 - ¹⁵ G. Strang, "On the construction and comparison of difference schemes" *SIAM J. Numer. Anal.* **5**, 506-517 (1968).
 - ¹⁶ Q. Sheng, "Solving linear partial differential equations by exponential splitting", *IMA Journal of numerical analysis*, **9**, 199-212 (1989).
 - ¹⁷ S. A. Chin, "Multi-product splitting and Runge-Kutta-Nystrom integrators", *Cele. Mech. Dyn. Astron.* **106**, 391-406 (2010).

- ¹⁸ S. A. Chin and Jurgen Geiser, “Multi-product operator splitting as a general method of solving autonomous and nonautonomous equations”, *IMA Journal of Numerical Analysis*, **31** 1552-1577 (2011); doi: 10.1093/imanum/drq022
- ¹⁹ M. Schatzman, “Numerical integration of reaction-diffusion systems”, *Numerical Algorithms* **31** 247-269 (2002).
- ²⁰ S. Blanes, F. Casas and J. Ros, *Extrapolation of symplectic integrators*, *Celest. Mech. Dyn. Astron.*, **75** (1999)149-161
- ²¹ W. Hundsdorfer and J. G. Verwer, *Numerical Solution of Time-Dependent Advection-Diffusion-Reaction Equations*, P.119, Springer-Verlag Berlin Heidelberg 2003.
- ²² R. E. Zillich, J. M. Mayrhofer and S. A. Chin, “Extrapolated high-order propagators for path integral Monte Carlo simulations”, *J. Chem. Phys.* **132**, 044103 (2010).
- ²³ K. V. Robert and N. O. Weiss, “Convective difference schemes”, *Math. Comput.* **20**, 272-299 (1966)
- ²⁴ L. J. Campbell and B. Yin, “On the stability of Alternating-Direction Explicit Methods for Advection-Diffusion Equations”, *Num. Methods for PDE* **23**, 1429-1444 (2007); DOI: 10.1002/num.20233
- ²⁵ R. I. McLachlan, “On the numerical integration of ordinary differential equations by symmetric composition methods”, *SIAM J. Sci. Comput.* **16**, 151 (1995).
- ²⁶ S. Blanes and P. C. Moan, “Practical symplectic partition Runge-Kutta methods and Runge-Kutta Nyström methods”, *J. Comput. Appl. Math.* **142**, 313 (2002).